**Resumen UT8**

Grafo no dirigido G = (V,A) consta de un conjunto finito de vértices V y un conjunto de aristas A. Se diferencia en que cada arista A es un par no ordenado de vértices (v, w) = (w, v).

Los vértices v y w son adyacentes si (v, w) es una arista o (w, v) lo es. Se dice que la arista (v, w) es incidente sobre los vértices v y w.

**Camino** --> secuencia de vértices v1, v2, … , vn tal que (vi, vi+1) es una arista para 1 =< i < n.

**Camino simple** --> camino con todos los vértices distintos, excepto primero y ultimo que pueden ser el mismo.

**La longitud del camino es n-1, el número de aristas a lo largo del camino.**

**Grafo conexo** --> todos los pares de vértices conectados.

Sea G = (V, A) un grafo, un subgrafo de G es un grafo G’ = (V’, A’) donde V’ es un subconjunto de V, y A’ consta de las aristas (v, w) en A tales que v y w están en A’.

Un componente conexo de un grafo G es un subgrafo conexo inducido maximal, esto es, un subgrafo conexo inducido que por sí mismo no es un subgrafo propio de ningún otro subgrafo conexo de G.

Un ciclo (simple) de un grafo es un camino (simple) de longitud mayor o igual a tres, que conecta un vértice consigo mismo. No se consideran ciclos los caminos de la forma v (camino de longitud 0), v, v (camino de longitud 1), o v, w, v (camino de longitud 2).

Un grafo es cíclico si contiene como mínimo 1 ciclo.

Grafo conexo cíclico (árbol libre), todo árbol libre con n >= 1 vértices tienen n – 1 aristas.

Si se agrega cualquier arista a un árbol libre se forma un ciclo.

**Métodos de representación** --> se usan los mismos métodos que para los árboles dirigidos.

La matriz de adyacencia resulta simétrica.

**Árboles abarcadores de costo mínimo**

Un árbol abarcador para un grafo G es un árbol libre que conecta todos los vértices de V, su costo es la suma de los costos de las aristas del árbol.

Si los vértices representan ciudades y las aristas líneas de comunicación entre ellas, un árbol abarcador de costo mínimo representa una red que comunica todas las ciudades a un costo minimal.

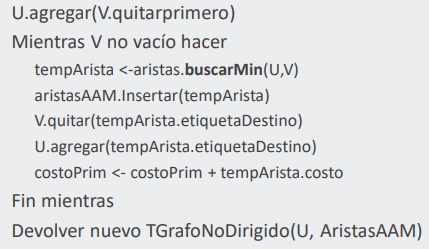
**La propiedad AAM** (Árbol abarcador de costo mínimo)

Sea G = (V, A) un grafo conexo con una función de costo definida en las aristas, sea U algún subconjunto propio de V. Si (u, v) es una arista de costo mínimo tal que u U y v V – U, existe un árbol abarcador de costo mínimo que incluye (u, v) en sus aristas.

**Algoritmo de Prim**

Suponga que V = (1, 2, … , n). El algoritmo de Prim comienza cuando se asigna a un conjunto U un valor inicial {1}, el cual crece en un árbol abarcador arista por arista. En cada paso localiza la arista más corta (u, v) que conecta U y V – U, a U. Este paso se repite hasta que U = V.

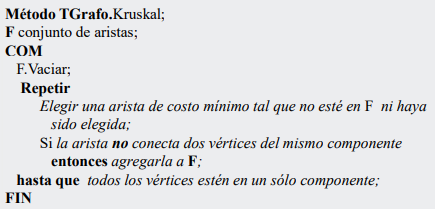
El costo de tiempo del algoritmo de Prim es O(n2). El algoritmo de Kruskal tiene costo O (a loga), donde a es el número de aristas del grafo. Si a es menor que n2 el algoritmo de Kruskal es mejor.





**Algoritmo de Kruskal**

Se empieza con un grafo T = (V , ) constituido por los vértices de G sin aristas (cada vértice es un componente conexo en sí mismo). Se examinan las aristas a partir de A en orden creciente de acuerdo con el costo. Si la arista conecta dos vértices en componentes distintos, entonces se agrega la arista a T. Se descartará si conecta dos vértices contenidos por el mismo componente. Cuando todos los vértices están en un solo componente, T es un árbol abarcador de costo mínimo para G.



Texto

Descripción generada automáticamente

**Recorridos**

En un gran número de problemas con grafos, es necesario visitar sistemáticamente los vértices del grafo. Las búsquedas en profundidad y en amplitud, son técnicas que pueden usarse para determinar de manera eficiente todos los vértices que están conectados a un vértice dado.

**Búsqueda en profundidad (bpf):**

Para grafo no dirigido solo hay 2 clases: aristas de árbol y aristas de retroceso.

Dado que en grafos no dirigidos no existe distinción entre las aristas de retroceso y las de avance se denominarán arcos de retroceso. En un grafo no dirigido no existen las aristas cruzadas, esto es, aristas (v, w) donde v no es antecesor ni descendiente de w en el árbol abarcador.

Como resultado, durante una búsqueda en profundidad en un grafo no dirigido G, todas las aristas pueden ser,

* Aristas de árbol, aquellas aristas (v, w) tales que bpf(v) llama directamente a bpf(w) o viceversa,
* Aristas de retroceso, aquellas aristas (v, w) tales que ni bpf(v) ni bpf(w) se llaman directamente, pero una llamó indirectamente a la otra (es decir, bpf(w) llama a bpf(x), que llama a bpf(v), de modo que w es antecesor de v).

**Búsqueda en amplitud:**

Cada vértice v que se visita se busca en forma tan amplia como sea posible, visitando todos los vértices adyacentes a v.

Toda arista que no es de árbol es una arista cruzada, conecta dos vértices que ninguno es antecesor de otro. Tiene mismo costo de ejecución que bpf O(a).

**Puntos de articulación y componentes biconexos**

Punto de articulación de un grafo es un vértice que cuando se elimina se divide un componente conexo en 2 o más partes. A un grafo sin puntos de articulación se le llama biconexo (la bpf es útil para encontrar componentes biconexos de un grafo). Un grafo tiene conectividad k si la eliminación de k-1 vértice cualquiera no lo desconecta.

La raíz es un punto de articulación si tiene 2 o más hijos.

Tiempo del algoritmo para encontrar puntos de articulación es O(n + a). Si n <=a es O(a).